



# **An Overview of the Thyra Interoperability Effort**

## **Current Status and Future Plans**

**Roscoe A. Bartlett**

**Department 1411: Optimization and Uncertainty Estimation**

**Sandia National Laboratories**

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company,  
for the United States Department of Energy under contract DE-AC04-94AL85000.





## Outline

---

- Introduction of abstract numerical algorithms (ANAs) and Trilinos software and interfaces
- The need for package interoperability and layering
- Thyra requirements and the foundational ANA operator/vector Interfaces
- History behind of Thyra
- Current status of Thyra
- Prototype and future Thyra
- Wrapping it up



## Outline

---

- Introduction of abstract numerical algorithms (ANAs) and Trilinos software and interfaces
- The need for package interoperability and layering
- Thyra requirements and the foundational ANA operator/vector Interfaces
- History behind of Thyra
- Current status of Thyra
- Prototype and future Thyra
- Wrapping it up



# Categories of Abstract Problems and Abstract Algorithms

## Trilinos Packages

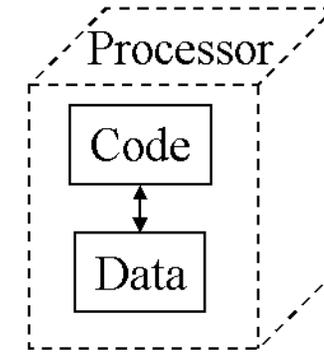
- **Linear Problems:** Given linear operator (matrix)  $A \in \mathbf{R}^{n \times n}$ 
  - **Linear equations:** Solve  $Ax = b$  for  $x \in \mathbf{R}^n$  Belos
  - **Eigen problems:** Solve  $Av = \lambda v$  for (all)  $v \in \mathbf{R}^n$  and  $\lambda \in \mathbf{R}$  Anasazi
  
- **Nonlinear Problems:** Given nonlinear operator  $f(x, p) \in \mathbf{R}^{n+m} \rightarrow \mathbf{R}^n$ 
  - **Nonlinear equations:** Solve  $f(x) = 0$  for  $x \in \mathbf{R}^n$  NOX
  - **Stability analysis:** For  $f(x, p) = 0$  find space  $p \in \mathcal{P}$  such that  $\frac{\partial f}{\partial x}$  is singular LOCA
  
- **Transient Nonlinear Problems:**
  - **DAEs/ODEs:** Solve  $f(\dot{x}(t), x(t), t) = 0, t \in [0, T], x(0) = x_0, \dot{x}(0) = x'_0$   
for  $x(t) \in \mathbf{R}^n, t \in [0, T]$  Rythoms
  
- **Optimization Problems:**
  - **Unconstrained:** Find  $p \in \mathbf{R}^m$  that minimizes  $g(p)$
  - **Constrained:** Find  $x \in \mathbf{R}^n$  and  $p \in \mathbf{R}^m$  that:  
minimizes  $g(x, p)$   
such that  $f(x, p) = 0$  MOOCHO



# Common Environments for Scientific Computing

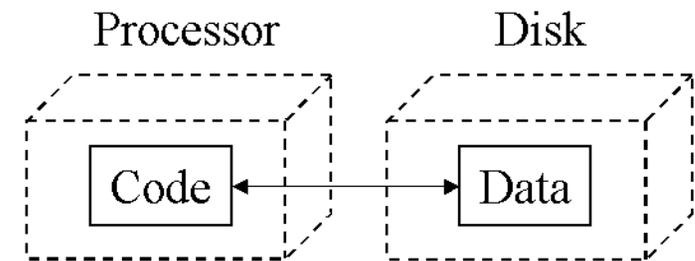
## Serial / SMP (symmetric multi-processor)

- All data stored in RAM in a single local process



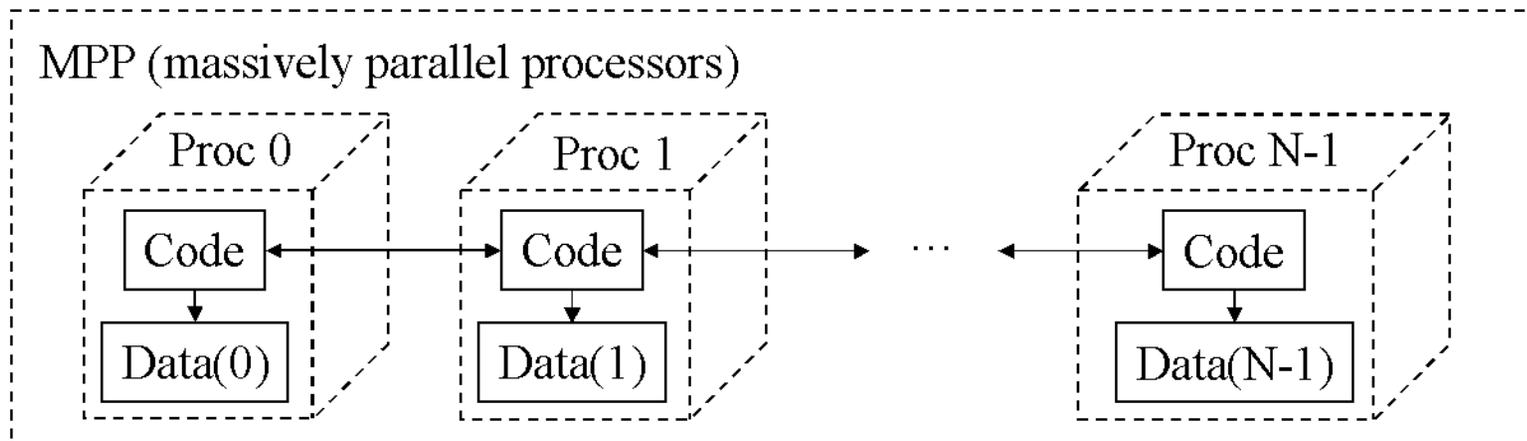
## Out of Core

- Data stored in file(s) (too big to fit in RAM)



## SPMD (Single program multiple data)

- Same code on each processor but different data





# Introducing Abstract Numerical Algorithms

## What is an abstract numerical algorithm (ANA)?

An ANA is a numerical algorithm that can be expressed abstractly solely in terms of vectors, vector spaces, and linear operators (i.e. not with direct element access)

### Example Linear ANA (LANA) : Linear Conjugate Gradients

Given:

$A \in \mathcal{X} \rightarrow \mathcal{X}$  : s.p.d. linear operator

$b \in \mathcal{X}$  : right hand side vector

Find vector  $x \in \mathcal{X}$  that solves  $Ax = b$

### Linear Conjugate Gradient Algorithm

### Types of operations   Types of objects

Compute  $r^{(0)} = b - Ax^{(0)}$  for the initial guess  $x^{(0)}$ .

for  $i = 1, 2, \dots$

$$\rho_{i-1} = \langle r^{(i-1)}, r^{(i-1)} \rangle$$

$$\beta_{i-1} = \rho_{i-1} / \rho_{i-2} \quad (\beta_0 = 0)$$

$$p^{(i)} = r^{(i-1)} + \beta_{i-1} p^{(i-1)} \quad (p^{(1)} = r^{(1)})$$

$$q^{(i)} = Ap^{(i)}$$

$$\gamma_i = \langle p^{(i)}, q^{(i)} \rangle$$

$$\alpha_i = \rho_{i-1} / \gamma_i$$

$$x^{(i)} = x^{(i-1)} + \alpha_i p^{(i)}$$

$$r^{(i)} = r^{(i-1)} - \alpha_i q^{(i)}$$

check convergence; continue if necessary

end

linear operator applications

vector-vector operations

Scalar operations

scalar product  $\langle x, y \rangle$  defined by vector space

Linear Operators

- $A$

Vectors

- $r, x, p, q$

Scalars

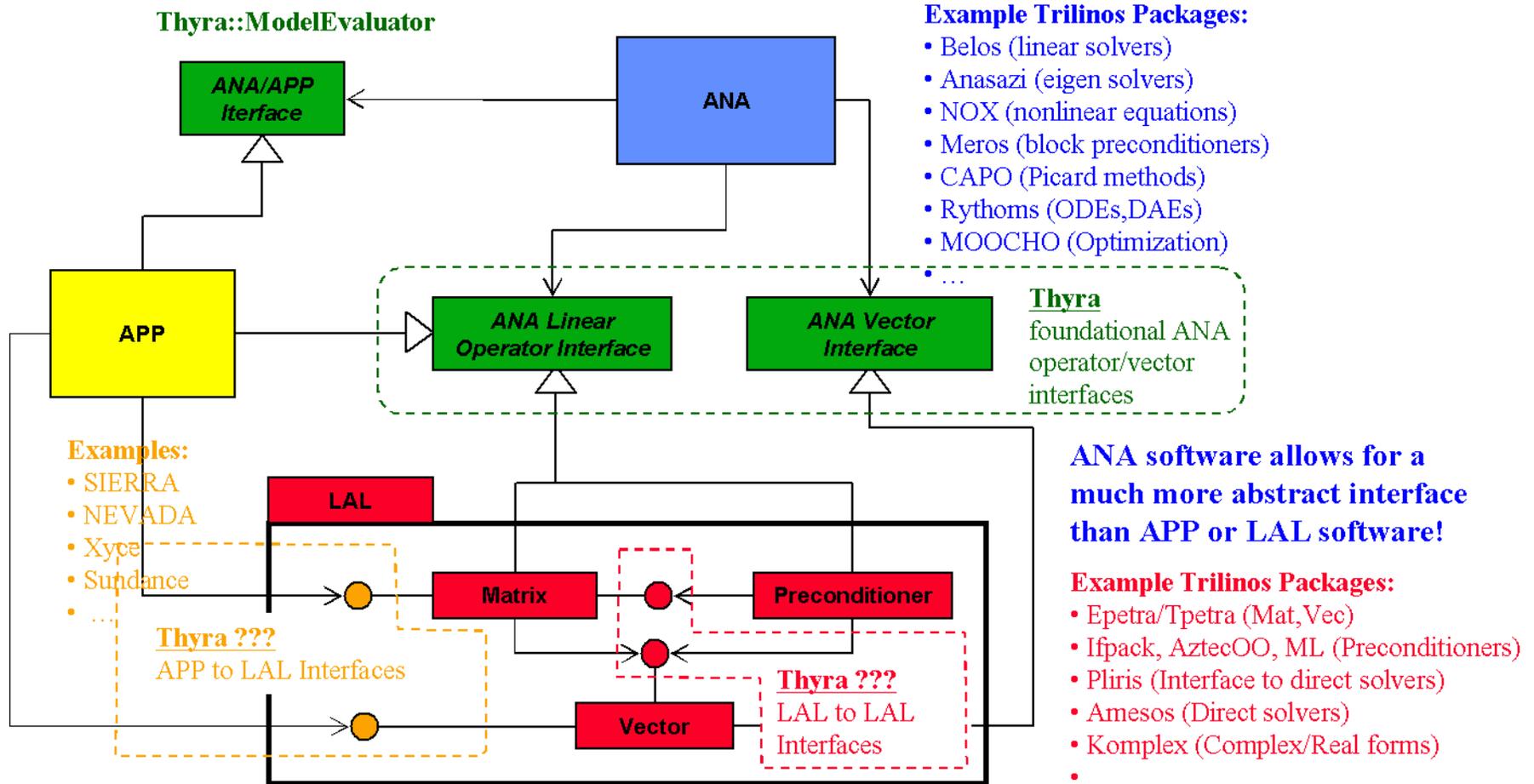
- $\rho, \beta, \gamma, \alpha$

Vector spaces?

- $\mathcal{X}$



# Software Componentization and Trilinos Interfaces



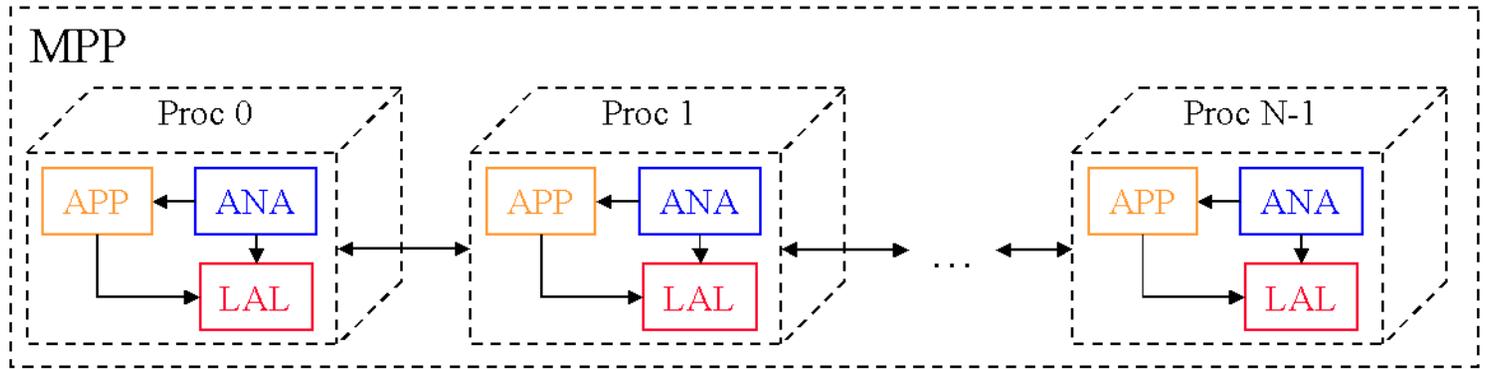
## Types of Software Components

- 1) **ANA** : Abstract Numerical Algorithm (e.g. linear solvers, eigen solvers, nonlinear solvers, stability analysis, uncertainty quantification, transient solvers, optimization etc.)
- 2) **LAL** : Linear Algebra Library (e.g. vectors, sparse matrices, sparse factorizations, preconditioners)
- 3) **APP** : Application (the model: physics, discretization method etc.)

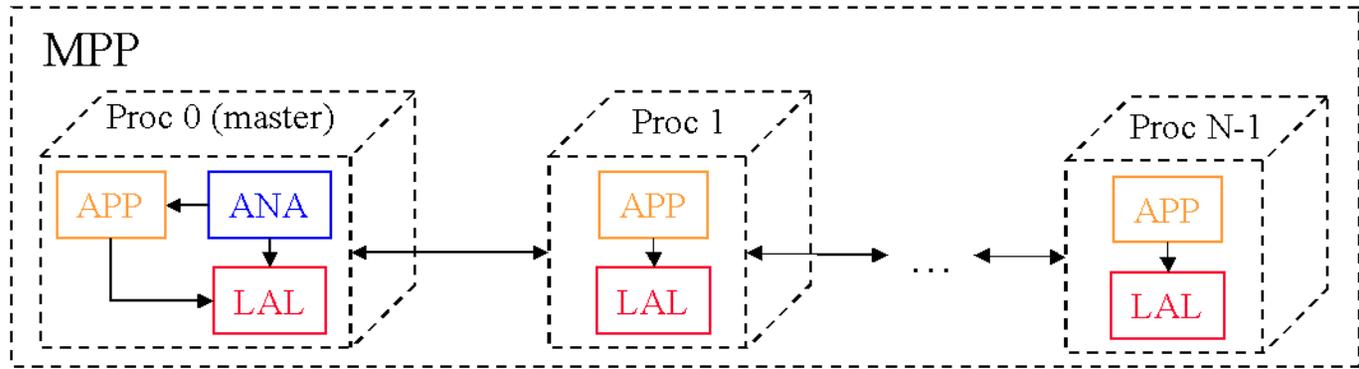


# Different Platform Configurations for Running ANAs

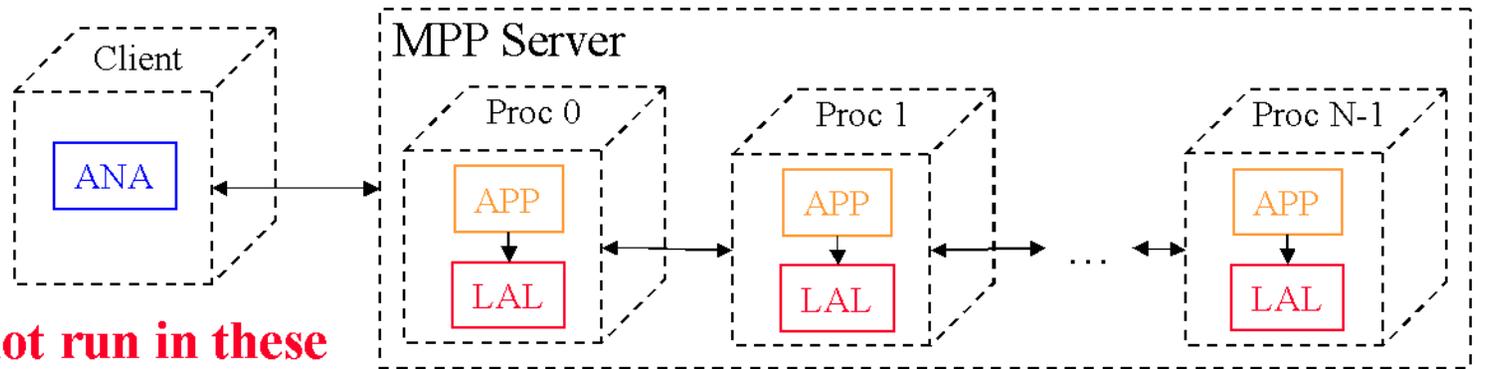
**SPMD**



**Master/Slave**



**Client/Server  
Master/Slave**



**If a code can not run in these modes it is not an ANA code!**



## Outline

---

- Introduction of abstract numerical algorithms (ANAs) and Trilinos software and interfaces
- The need for package interoperability and layering
- Thyra requirements and the foundational ANA operator/vector Interfaces
- History behind of Thyra
- Current status of Thyra
- Prototype and future Thyra
- Wrapping it up



## Trilinos Strategic Goals

---

- **Scalable Solvers**: As problem size and processor counts increase, the cost of the solver will remain a nearly fixed percentage of the total solution time.
- **Hardened Solvers**: Never fail unless problem essentially unsolvable, in which case we diagnose and inform the user why the problem fails and provide a reliable measure of error.
- **Full Vertical Coverage**: Provide leading edge capabilities from basic linear algebra to transient and **optimization solvers**.
- **Grand Universal Interoperability**: All Trilinos **packages** will be interoperable, so that any combination of solver packages that makes sense algorithmically will be **possible** within Trilinos.
- **Universal Solver RAS**: Trilinos will be:
  - Integrated into every major application at Sandia (Availability).
  - The leading edge hardened, scalable solutions for each of these applications (Reliability).
  - Easy to maintain and upgrade within the application environment (Serviceability).

**Thyra** is being developed to address this issue

Courtesy of Mike Heroux, Trilinos Project Leader

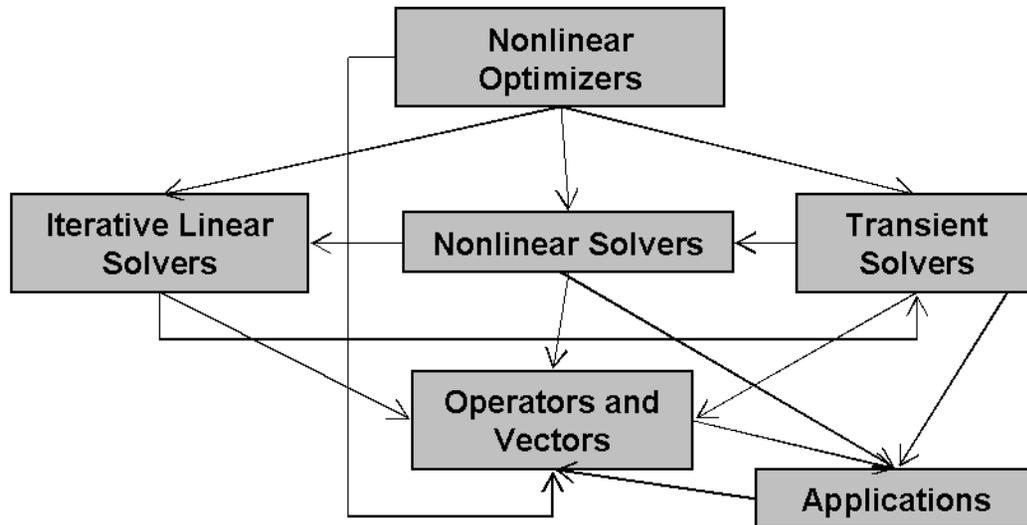
### Key Point

- Universal Interoperability will not happen automatically and if not done carefully then can compromise the other strategic goals



## Interoperability is Especially Important to Optimization

Numerous interactions exist between abstract numerical algorithms (ANAs) in a transient optimization problem



What is needed to solve problem?

- Standard interfaces to break  $O(N^2)$ 
  - 1-to-1 couplings
    - Operators/vectors
    - Linear Solvers
    - Nonlinear solvers
    - Transient solvers
    - etc.

**This is hard!** This level of interoperability for massively parallel algorithms has never been achieved before!

### Key Points

- Higher level algorithms, like optimization, require a lot of interoperability
- Interoperability must be “easy” or these configurations will not be possible
- Many real problems even more complicated



# Example of Algorithm Interoperability and Layering : Rythmos

Solve  $f(\dot{x}(t), x(t), t) = 0, t \in [t_0, t_f], x(t_0) = x_0, \dot{x}(t_0) = \dot{x}_0$   
for  $x(t) \in \mathbf{R}^n, t \in [t_0, t_f]$

## Time Stepper

Advance  $x(t_k)$  to  $x(t_{k+1})$   
where  $t_{k+1} = t_k + \Delta t_k$

## Implicit Backward Euler method

Solve  $f\left(\frac{x_{k+1}-x_k}{\Delta t_k}, x_{k+1}, t_{k+1}\right) = 0$  for  $x_{k+1}$

## Nonlinear equations

Solve  $r(z) = 0$  for  $z \in \mathbf{R}^n$

## Newton's method (e.g. NOX)

Choose initial guess  $z_0$ , tolerance  $\eta$   
**for**  $k = 0, 1, \dots$

If "converged" **Stop!**

Solve  $\frac{\partial r(z_k)}{\partial z} \delta z_k = -r(z_k)$  for  $\delta z$

Choose  $\alpha$  using a line search method

$z_{k+1} = z_k + \alpha \delta z_k$

**end for**

## Linear equations

Solve  $Ax = b$  for  $x \in \mathbf{R}^n$

## Preconditioned GMRES

Iterate to "convergence"

$PAx = Pb$

## Operator and Preconditioner applications

Apply  $y = Ax$  **App Defined?**

Apply  $y = Px$  **App Defined?**

Preconditioners can be defined in many different ways



## Outline

---

- Introduction of abstract numerical algorithms (ANAs) and Trilinos software and interfaces
- The need for package interoperability and layering
- Thyra requirements and the foundational ANA operator/vector Interfaces
- History behind of Thyra
- Current status of Thyra
- Prototype and future Thyra
- Wrapping it up



## Requirements for Abstract Numerical Algorithms and Thyra

---

**An important consideration**  $\Rightarrow$  **Scientific computing is computationally expensive!**

### **Abstract Interfaces for Abstract Numerical Algorithms using Thyra must:**

- Be portable to all ASC (advanced scientific computing) computer platforms
- Provide for stable and accurate numerics
- Result in algorithms with near-optimal storage and runtime performance
  - Scientific computing is expensive!

**An important ideal**  $\Rightarrow$  **A customized hand-code algorithm in Fortran 77 should not provide significant improvements in storage requirements, speed or numerical stability!**

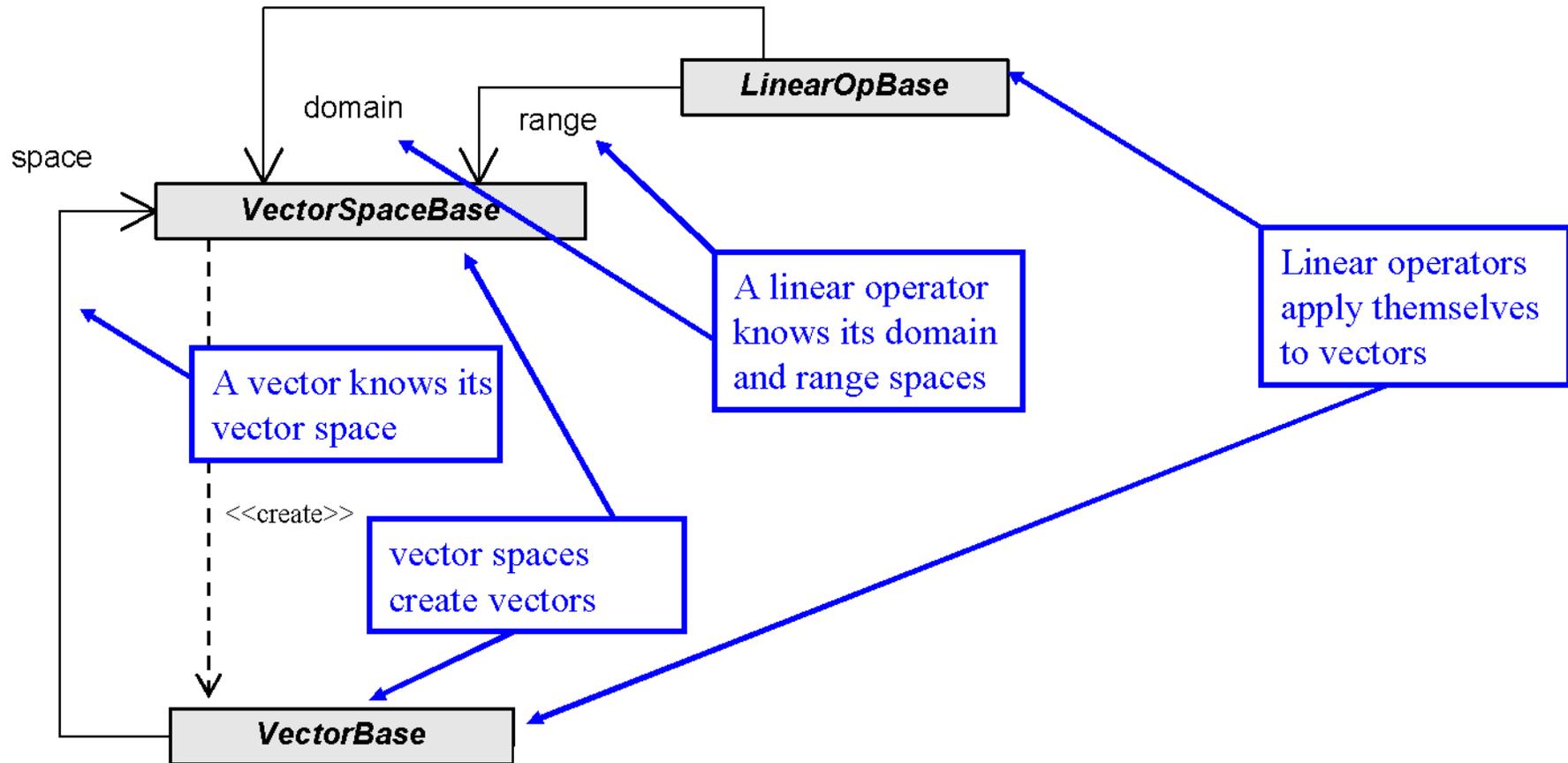
**An important ideal**  $\Rightarrow$  **Object-oriented “overhead” should be constant and not increase as the problem size increases.**

### **Abstract Interfaces for Abstract Numerical Algorithms using Thyra should:**

- Be minimal but complete (good object-oriented design principle)
- Support a variety of computing platforms and configurations
  - i.e. Serial/SMP, Out-of-Core, SPMD, master/slave and client/server
- Be (relatively) easy to provide new implementations **(Subjective!!!)**
- Not be too complicated **(Subjective!!!)**



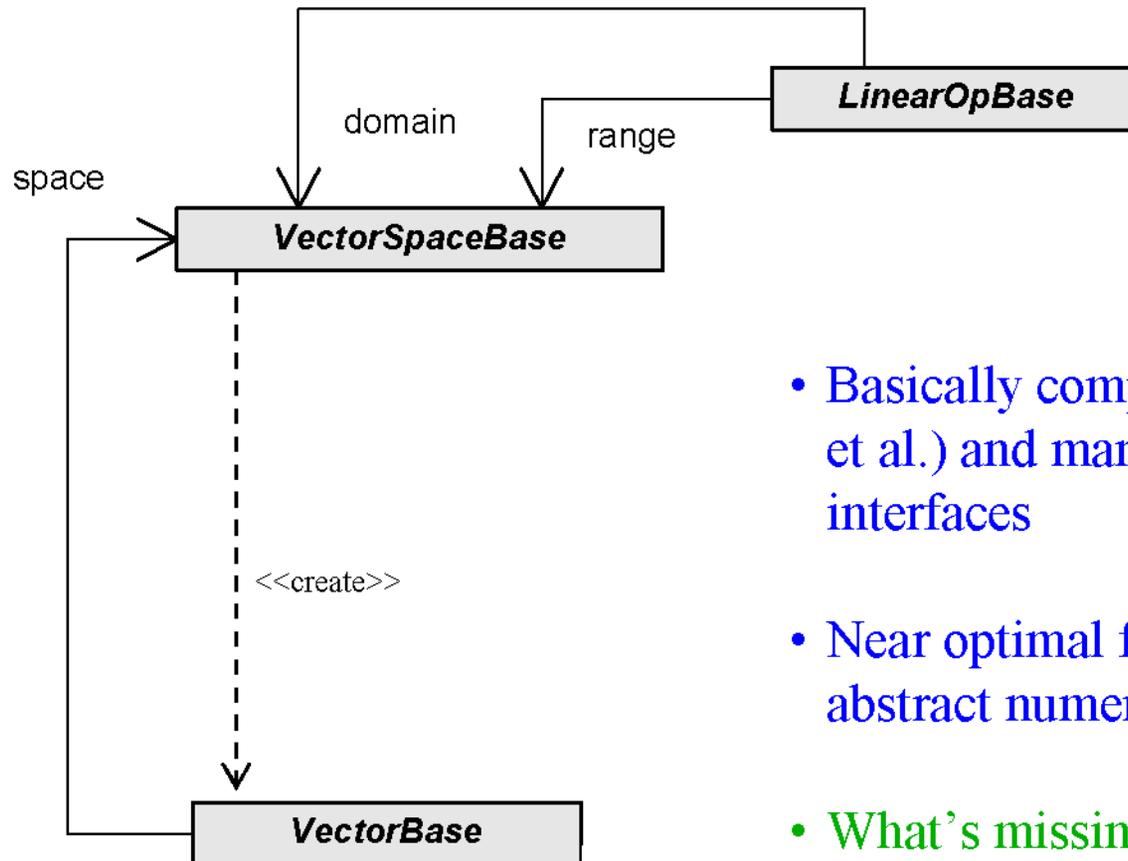
# Foundational Thyra ANA Operator/Vector Interfaces



**Unified Modeling Language (UML) Notation!**



## Foundational Thyra ANA Operator/Vector Interfaces



- Basically compatible with HCL (Symes et al.) and many other operator/vector interfaces
- Near optimal for many but not all abstract numerical algorithms (ANAs)
- What's missing?

=> Multi-vectors!



# Introducing Multi-Vectors

## What is a multi-vector?

- An  $m$  multi-vector  $V$  is a tall thin dense matrix composed of  $m$  column vectors  $v_j$

$$V = \begin{bmatrix} v_1 & v_2 & \dots & v_m \end{bmatrix} \in \mathcal{S} \times \mathbf{R}^m$$

Example:  $m = 4$  columns

$$V = \begin{bmatrix} | & | & | & | \\ | & | & | & | \\ | & | & | & | \\ | & | & | & | \end{bmatrix} = \begin{bmatrix} | & | & | & | \\ | & | & | & | \\ | & | & | & | \\ | & | & | & | \end{bmatrix}$$

## What ANAs can exploit multi-vectors?

- Block linear solvers (e.g. block GMRES)
- Block eigen solvers (i.e. block Arnoldi)
- Compact limited memory quasi-Newton
- Tensor methods for nonlinear equations

## Why are multi-vectors important?

- Cache performance
- Reduce global communication

## Examples of multi-vector operations

- Block update

$$Y = Y + X Q$$

- Operator applications (i.e. mat-vecs)

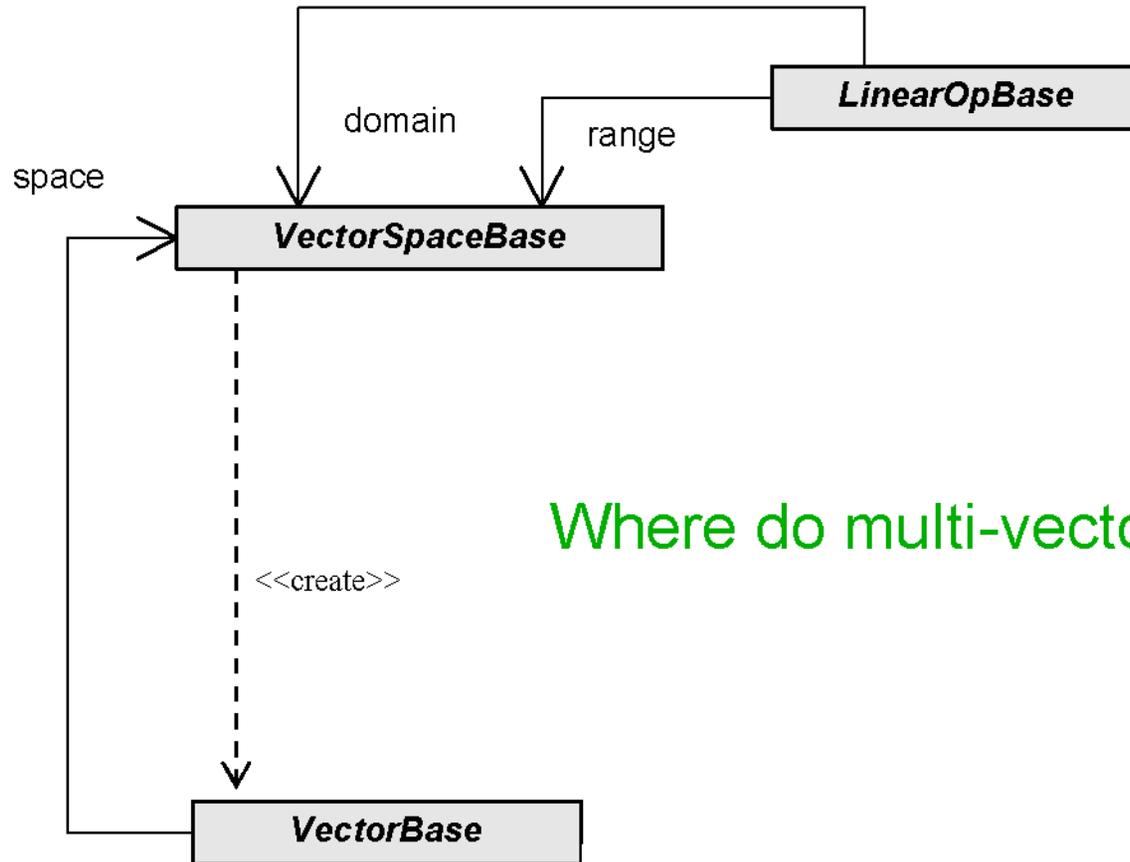
$$Y = A X$$

- Block dot products ( $m^2$ )

$$Q = X^T Y$$



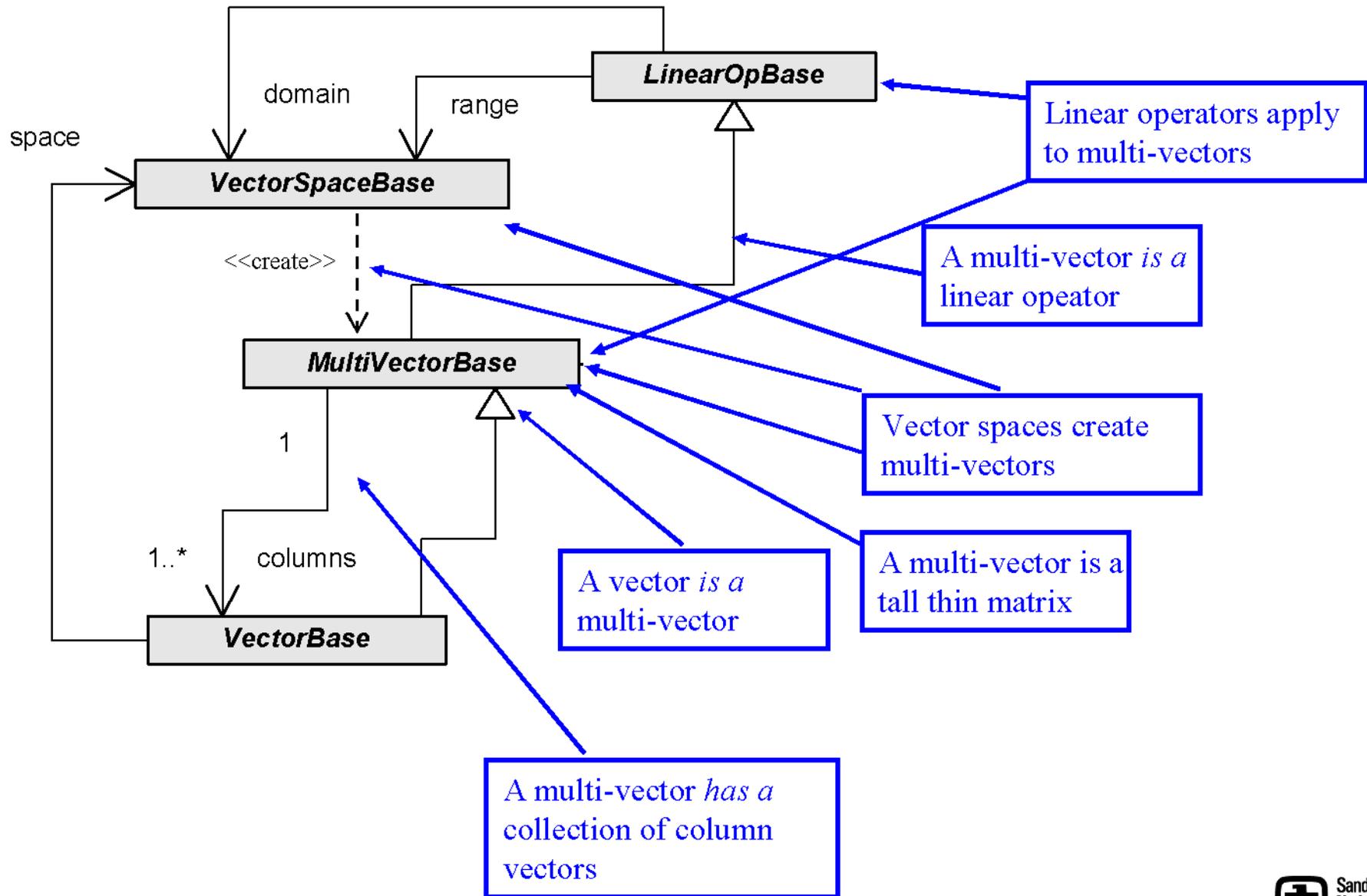
# Foundational Thyra ANA Operator/Vector Interfaces



Where do multi-vectors fit in?

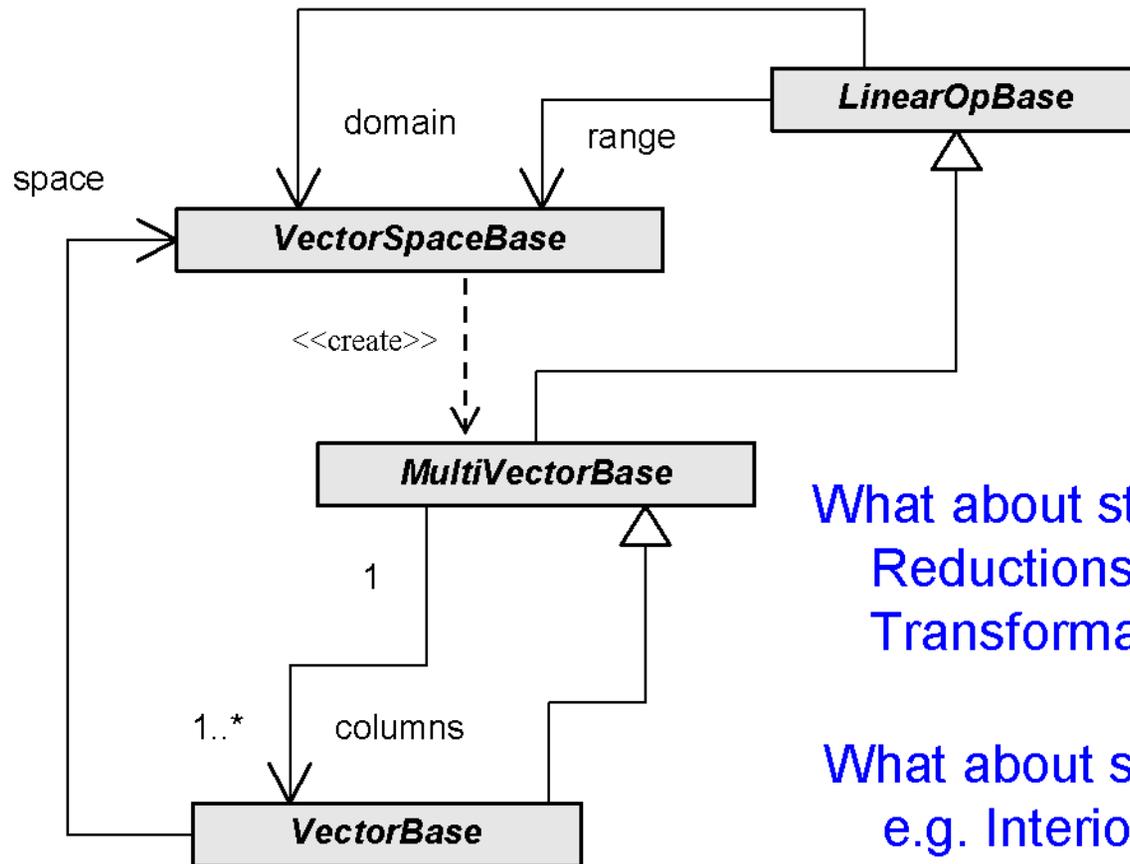


# Foundational Thyra ANA Operator/Vector Interfaces





# Foundational Thyra ANA Operator/Vector Interfaces



What about standard vector ops?  
Reductions (norm, dot etc.)?  
Transformations (axpy, scaling etc.)?

What about specialized vector ops?  
e.g. Interior point methods for opt



# What's the Big Deal about Vector-Vector Operations?

## Examples from OOQP (Gertz, Wright)

$$y_i \leftarrow y_i + \alpha x_i z_i, i = 1 \dots n$$

$$y_i \leftarrow y_i / x_i, i = 1 \dots n$$

$$y_i \leftarrow \begin{cases} y^{\min} - y_i & \text{if } y_i < y^{\min} \\ y^{\max} - y_i & \text{if } y_i > y^{\max} \\ 0 & \text{if } y^{\min} \leq y_i \leq y^{\max} \end{cases}, i = 1 \dots n$$

$$\alpha \leftarrow \{ \max \alpha : x + \alpha d \geq \beta \}$$

## Example from TRICE (Dennis, Heinkenschloss, Vicente)

$$d_i \leftarrow \begin{cases} (b - u)_i^{1/2} & \text{if } w_i < 0 \text{ and } b_i < +\infty \\ 1 & \text{if } w_i < 0 \text{ and } b_i = +\infty \\ (u - a)_i^{1/2} & \text{if } w_i \geq 0 \text{ and } a_i > -\infty \\ 1 & \text{if } w_i \geq 0 \text{ and } a_i = -\infty \end{cases}, i = 1 \dots n$$

Many different and unusual vector operations are needed by interior point methods for optimization!

## Example from IPOPT (Wächter)

$$x_i \leftarrow \begin{cases} \left( x_i^L + \frac{(x_i^U - x_i^L)}{2} \right) & \text{if } \hat{x}_i^L > \hat{x}_i^U \\ \hat{x}_i^L & \text{if } x_i < \hat{x}_i^L \\ \hat{x}_i^U & \text{if } x_i > \hat{x}_i^U \end{cases}, i = 1 \dots n$$

Currently in MOOCHO :  
> 40 vector operations!

where :

$$\hat{x}_i^L = \min \left( x_i^L + \eta (x_i^U - x_i^L), x_i^L + \delta \right)$$
$$\hat{x}_i^U = \max \left( x_i^L - \eta (x_i^U - x_i^L), x_i^U - \delta \right)$$



# Vector Reduction/Transformation Operators Defined

## Reduction/Transformation Operators (RTOp) Defined

$z^1 \dots z^q \leftarrow \text{op}_t(i, v^1 \dots v^p, z^1 \dots z^q)$       element-wise transformation  
 $\beta \leftarrow \text{op}_r(i, v^1 \dots v^p, z^1 \dots z^q)$       element-wise reduction  
 $\beta^2 \leftarrow \text{op}_{rr}(b^1, b^2)$       reduction of intermediate reduction objects

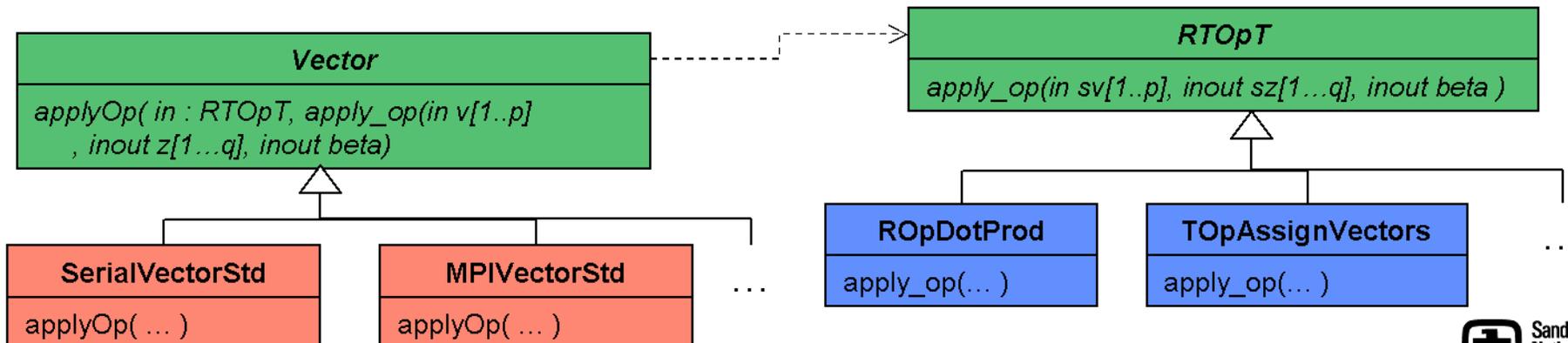
- $v^1 \dots v^p \in \mathbf{R}^n$  :  $p$  non-mutable (constant) input vectors
- $z^1 \dots z^q \in \mathbf{R}^n$  :  $q$  mutable (non-constant) input/output vectors
- $\beta$  : reduction target object (many be non-scalar (e.g.  $\{y_k, k\}$ ), or NULL)

## Key to Optimal Performance

- $\text{op}_t(\dots)$  and  $\text{op}_r(\dots)$  applied to entire sets of subvectors ( $i = a \dots b$ ) independently:  
 $z^1_{a:b} \dots z^q_{a:b}, \beta \leftarrow \text{op}(a, b, v^1_{a:b} \dots v^p_{a:b}, z^1_{a:b} \dots z^q_{a:b}, \beta)$
- Communication between sets of subvectors only for  $\beta^1$  NULL,  $\text{op}_{rr}(\beta^1, \beta^2) \rightarrow \beta^2$

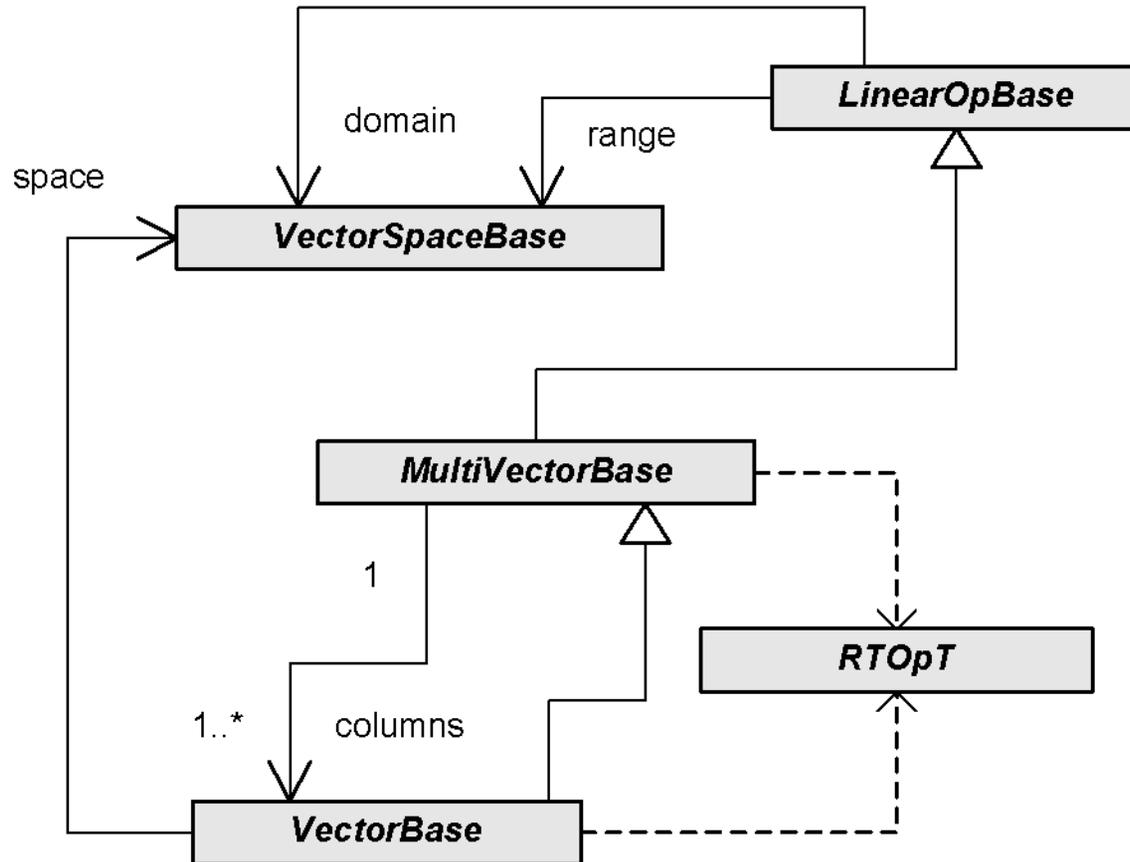
## Software Implementation (see RTOp paper on Trilinos website "Publications")

- "Visitor" design pattern (a.k.a. function forwarding)





# Foundational Thyra ANA Operator/Vector Interfaces



## The Key to success! Reduction/Transformation Operators

- Supports all needed vector operations
- Data/parallel independence
- Optimal performance

R. A. Bartlett, B. G. van Bloemen Waanders and M. A. Heroux. Vector Reduction/Transformation Operators, ACM TOMS, March 2004



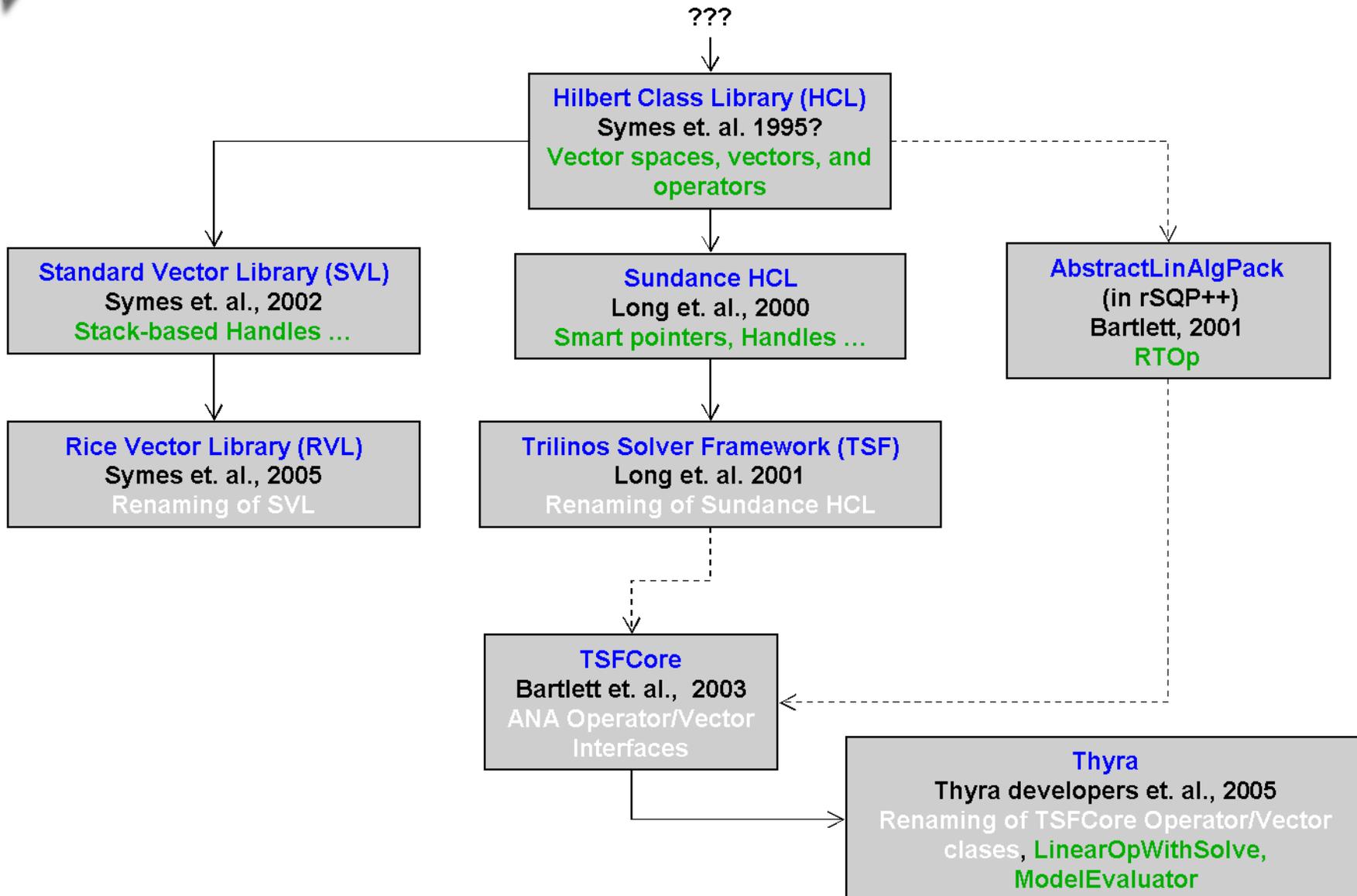
## Outline

---

- Introduction of abstract numerical algorithms (ANAs) and Trilinos software and interfaces
- The need for package interoperability and layering
- Thyra requirements and the foundational ANA operator/vector Interfaces
- History behind of Thyra
- Current status of Thyra
- Prototype and future Thyra
- Wrapping it up



# History for Thyra and Related Software





## Outline

---

- Introduction of abstract numerical algorithms (ANAs) and Trilinos software and interfaces
- The need for package interoperability and layering
- Thyra requirements and the foundational ANA operator/vector Interfaces
- History behind of Thyra
- **Current status of Thyra**
- Prototype and future Thyra
- Wrapping it up



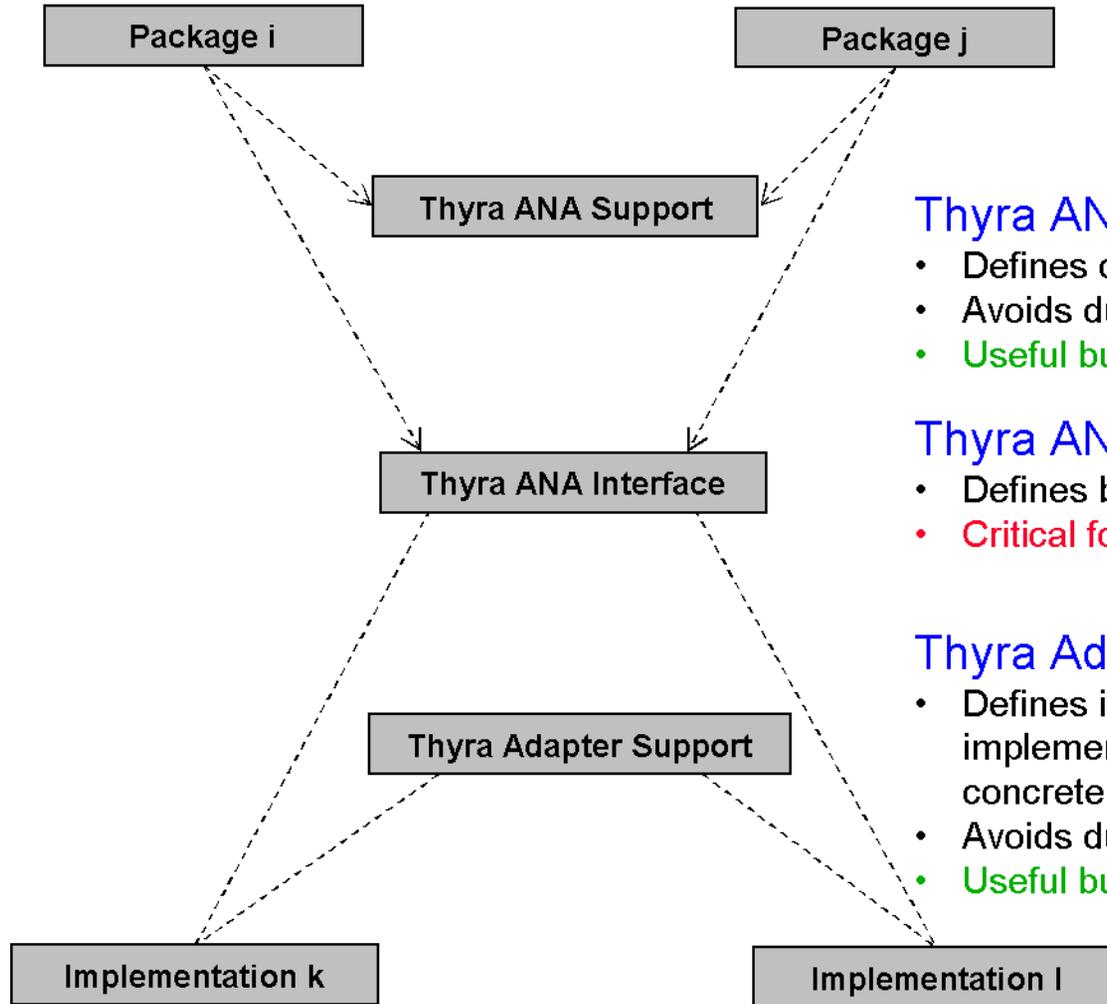
## Outline

---

- Introduction of abstract numerical algorithms (ANAs) and Trilinos software and interfaces
- The need for package interoperability and layering
- Thyra requirements and the foundational ANA operator/vector Interfaces
- History behind of Thyra
- Current status of Thyra
  - Three use cases and the scope of Thyra
  - Assorted ANA support software
- Prototype and future Thyra
- Wrapping it up



# Three Different Types of Use Cases for Thyra ANA Interfaces



## Thyra ANA Support Software

- Defines conveniences to aid in writing ANAs
- Avoids duplication of effort
- Useful but optional!

## Thyra ANA Interoperability Interfaces

- Defines basic functionality needed for ANAs
- Critical for scalable interoperability!

## Thyra Adapters Support Software

- Defines infrastructure support and concrete implementations to make it easy to provide concrete implementations for Thyra ANA interfaces
- Avoids duplication of effort
- Useful but optional!



## Three Use Cases for Fundamental Thyra Interfaces

---

1. **Thyra as interoperability layer for linear ANA objects (i.e. linear operators, vectors, multi-vectors, reduction/transformation operators)**
2. **Development of concrete Thyra implementation subclasses (“Adapters Support”)**
  1. Support for decoupling application-defined scalar products from concrete vector spaces
  2. Support for concrete implementations
    1. Support and concrete subclasses for serial shared-memory platforms
    2. Support and concrete subclasses for MPI SPMD distributed memory platforms
    3. Wrappers (i.e. adapters) for Epetra objects
3. **Using Thyra objects for the development of ANAs (“ANA Support”)**
  1. C++ wrapper functions for prewritten RTOpT subclasses  
e.g. `norm(x)`, `assign(&y,x)` ... [See documentation]
  2. Composite objects:
    1. Product vectors, multi-vectors and vector spaces  
e.g. `x = [ v1; v2; v3; ... ; vn ];` [See documentation]
    2. Decorator and composite linear operators
      1. Implicit adjoint/transpose: e.g. `M = adjoint(A);` [ScaledAdjointLinearOp]
      2. Additive linear operators: e.g. `M = A + B + C + D;` [AdditiveLinearOp]
      3. Multiplicative linear operators: e.g. `M = A * B * C * D;` [MultiplicativeLinearOp]
      4. Block linear linear operators: e.g. `M = [ A, B ; C, D ];` [BlockLinearOp]
  3. Handle classes for operator overloading (currently in TSFExtended)  
e.g. `y = alpha*trans(A)*x + gamma*B*z + beta*y`

[Trilinos Thyra Website](#)



## Outline

---

- Introduction of abstract numerical algorithms (ANAs) and Trilinos software and interfaces
- The need for package interoperability and layering
- Thyra requirements and the foundational ANA operator/vector Interfaces
- History behind of Thyra
- Current status of Thyra
  - Three use cases and the scope of Thyra
  - Assorted ANA support software
- Prototype and future Thyra
- Wrapping it up



# Thyra Composite/Decorator ANA Operator/Vector Subclasses

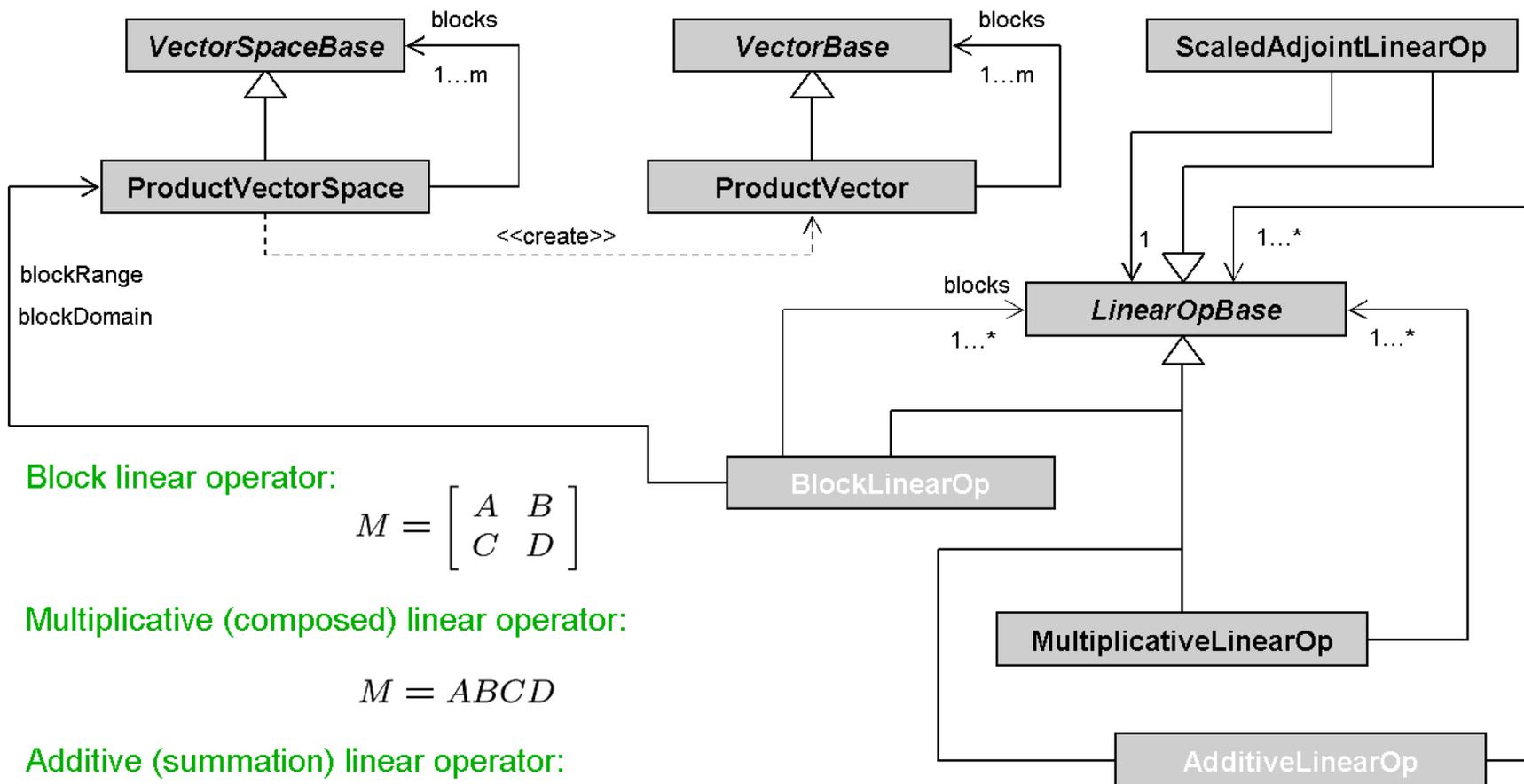
“Composite” subclasses allow a collection of objects to be manipulated as one object

- Product vector spaces and product vectors:
  - Product vector spaces:  $\mathcal{X} = \mathcal{V}_1 \times \mathcal{V}_2 \times \dots \times \mathcal{V}_m$
  - Product vectors:  $x^T = [v_1^T \ v_2^T \ \dots \ v_m^T]$

“Decorator” subclasses wrap an object and changes its behavior

- Scaled/Adjoint(transposed) linear operator:

$$M = \alpha A^H$$



- Block linear operator:

$$M = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

- Multiplicative (composed) linear operator:

$$M = ABCD$$

- Additive (summation) linear operator:

$$M = A + B + C + D$$



## LinearOpTester Utility Class

```
template<class RangeScalar, class DomainScalar>
class LinearOpTester {
public:
    ...
    bool Thyra::LinearOpTester< RangeScalar, DomainScalar >::check (
        const LinearOpBase< RangeScalar, DomainScalar > &op
        ,MultiVectorRandomizerBase< RangeScalar > *rangeRandomizer
        ,MultiVectorRandomizerBase< DomainScalar > *domainRandomizer,
        ,std::ostream *out
        ,const std::string &leadingIndent = ""
        ,const std::string &indentSpacer = " "
        ) const;
    ...
};
```

- Given a LinearOpBase object, using randomly generated vector, check:
  - Is the forward operator truly linear?
  - If the adjoint is implemented:
    - Is the adjoint operator truly linear?
    - Is the adjoint operator consistent with the forward operator?
  - If checking for symmetry:
    - Is the adjoint the same as the forward operator?
- Highly configurable
  - Level of output varies from minimal pass/fail to dumping all vectors
  - Random vectors can be generated using strategy objects
  - All tests are governed by:
    - A flag for if the test is performed
    - Error and warning relative tolerance
  - 14 options in total



## LinearOpTester Utility Class : Example Test Output

```
this->check_linear_properties()==true: Checking the linear properties of the forward linear operator ...
```

```
op.applySupports(NONCONJ_ELE) == true ? passed
```

```
Checking that the forward operator is truly linear:
```

```
0.5*op*(v1 + v2) == 0.5*op*v1 + 0.5*op*v2
```

$\underbrace{\hspace{2em}}_{v3} \qquad \underbrace{\hspace{2em}}_{v5}$

$\underbrace{\hspace{4em}}_{v4} \qquad \underbrace{\hspace{4em}}_{v5}$

```
sum(v4) == sum(v5)
```

```
Random vector tests = 1
```

```
v1 = randomize(-1,+1); ...
```

```
v2 = randomize(-1,+1); ...
```

```
v3 = v1 + v2 ...
```

```
v4 = 0.5*op*v3 ...
```

```
v5 = op*v1 ...
```

```
v5 = 0.5*op*v2 + 0.5*v5 ...
```

```
Check: rel_err(sum(v4),sum(v5))  
= rel_err(-9.048598e-01,-9.048598e-01) = 1.104260e-15  
<= linear_properties_error_tol() = 1.000000e-04 : passed
```

- See test program `sillyCgSolve_serial.exe` for example



## Outline

---

- Introduction of abstract numerical algorithms (ANAs) and Trilinos software and interfaces
- The need for package interoperability and layering
- Thyra requirements and the foundational ANA operator/vector Interfaces
- History behind of Thyra
- Current status of Thyra
- **Prototype and future Thyra**
- Wrapping it up



## Outline

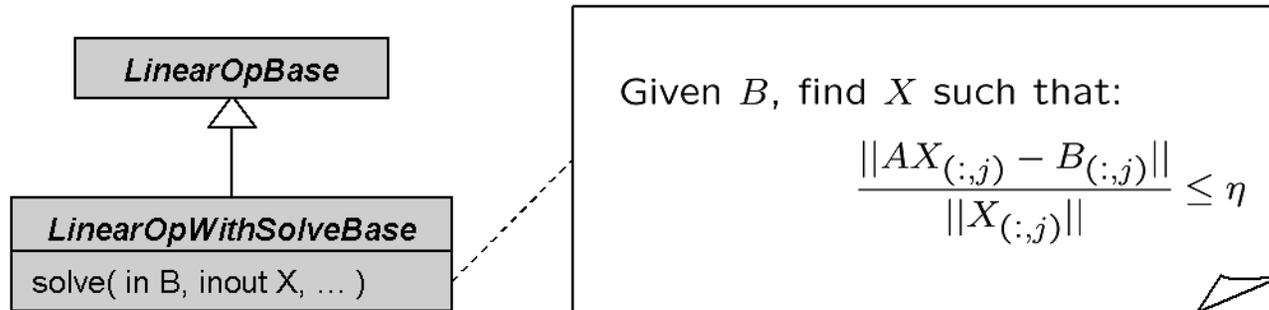
---

- Introduction of abstract numerical algorithms (ANAs) and Trilinos software and interfaces
- The need for package interoperability and layering
- Thyra requirements and the foundational ANA operator/vector Interfaces
- History behind of Thyra
- Current status of Thyra
- **Prototype and future Thyra**
  - **Linear solves, preconditioners, conversion from composed to explicit operators**
  - Nonlinear model evaluator
- Wrapping it up



# Prototype: Linear Operator With Solve and Factories

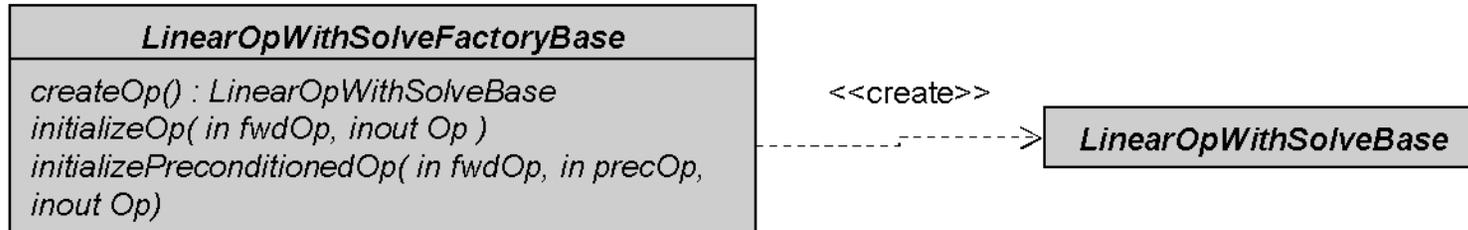
LinearOpWithSolveBase : Combines a linear operator and a linear solver



- Appropriate for both direct and iterative solvers
- Supports multiple simultaneous solutions as multi-vectors
- Allows targeting of different solution criteria to different RHSs
- Supports a “default” solve

LinearOpWithSolveFactoryBase :

- Uses LinearOpBase objects in initialize LinearOpWithSolveBase objects

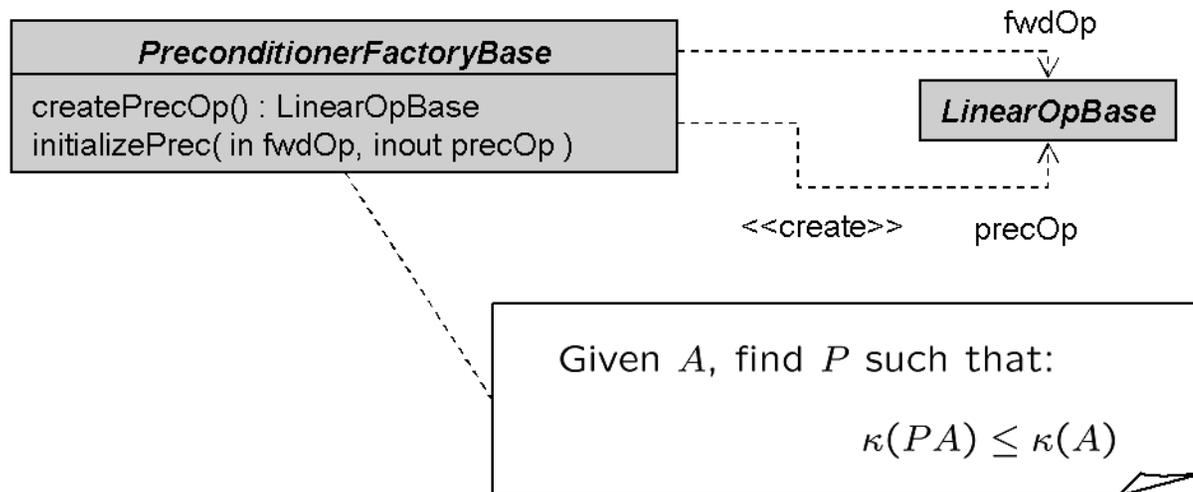


- Allows unlimited creation/reuse of LinearOpWithSolveBase objects
- Supports reuse of factorizations/preconditioners
- Supports client-created external preconditioners (which are ignored by direct solvers)
- Appropriate for both direct and iterative solvers
- Concrete adapters for Amesos and AztecOO available (Belos coming soon)



## Future Work: Preconditioners and Preconditioner Factories

PreconditionerFactoryBase : Creates and initializes preconditioner objects



- Allows unlimited creation/reuse of preconditioner objects
  - Supports reuse of factorization structures
  - Will have adapters for Ifpack, ML, etc ...
- Use Cases:
- Concrete iterative LinearOpWithSolveFactoryBase subclasses
    - AztecOO/Thyra adapters
    - Belos/Thyra adapters



## Future Work: Composed Operators and Explicit Operator Generation

---

- Implicitly composed operators: Combine blocked, added, multiplied and adjoint operations

- **Example:**

$$M = \left[ \begin{array}{c} \left[ \begin{array}{cc} \gamma AB + C & DE^H \\ G - A & P \end{array} \right]^H \\ \left[ LM \quad Q \right] \end{array} \right] \beta \left[ \begin{array}{c} I - J \\ A^H \\ W \end{array} \right]$$

- Strategy interface for converting implicitly composed operators into “explicit” operators

$$M \Rightarrow M_{\text{explicit}}$$

- “Explicit” operator can then be used to generate preconditioners or **LinearOpWithSolveBase** objects using appropriate factory objects
- Only applicable for matrix-based operators

- **Use Cases:**

- Certain Schur-complement preconditioners
- Certain optimization algorithms
- Meros (V. Howle)
- ???



## Outline

---

- Introduction of abstract numerical algorithms (ANAs) and Trilinos software and interfaces
- The need for package interoperability and layering
- Thyra requirements and the foundational ANA operator/vector Interfaces
- History behind of Thyra
- Current status of Thyra
- **Prototype and future Thyra**
  - Linear solves, preconditioners, conversion from composed to explicit operators
  - **Nonlinear model evaluator**
- Wrapping it up



# Prototype: Nonlinear Model Evaluator

**Motivation:** An interface for nonlinear problems is needed that will support a variety of different types of problems

- Nonlinear equations (and sensitivities)
- Stability analysis
- Explicit ODEs (and sensitivities)
- DAEs and implicit ODEs (and sensitivities)
- Unconstrained optimization
- Constrained optimization
- Uncertainty quantification
- ...

as well as different combinations of problem types such as:

- Uncertainty in transient simulations
- Stability of an optimum under uncertainty of a transient problem

**Key Point**

The number of combinations of different problem types is large and trying to statically type all of the combinations is not realistic

**Approach:** Develop a single, scalable interface to address all of these problems

• (Some) Input arguments:

- State and differential state:
- Parameter sub-vectors:
- Time (differential):

$$x \in \mathcal{X} \text{ and } \dot{x} = \frac{dx}{dt} \in \mathcal{X}$$

$$p_l \in \mathcal{P}_l \text{ for } l = 1 \dots N_p$$

$$t \in \mathbf{R}$$

**Key Point**

All inputs and outputs are optional and the model evaluator object itself decides which ones are accepted.

• (Some) Output functions:

- State function:
- Auxiliary response functions:
- State/state derivative operator (LinearOpWithSolve):

$$(\dot{x}, x, \{p_l\}, t) \Rightarrow f \in \mathcal{F}$$

$$(\dot{x}, x, \{p_l\}, t) \Rightarrow g_j \in \mathcal{G}_j, \text{ for } j = 1 \dots N_g$$

$$(\dot{x}, x, \{p_l\}, t) \Rightarrow W = \alpha \frac{\partial f}{\partial \dot{x}} + \beta \frac{\partial f}{\partial x}$$



## Model Evaluator : Some Examples of Nonlinear Problems

---

Nonlinear equations:	Solve $f(x) = 0$ for $x \in \mathbf{R}^n$
Stability analysis:	For $f(x, p) = 0$ find space $p \in \mathcal{P}$ such that $\frac{\partial f}{\partial x}$ is singular
Explicit ODEs:	Solve $\dot{x} = f(x, t) = 0, t \in [0, T], x(0) = x_0,$ for $x(t) \in \mathbf{R}^n, t \in [0, T]$
DAEs/Implicit ODEs:	Solve $f(\dot{x}(t), x(t), t) = 0, t \in [0, T], x(0) = x_0, \dot{x}(0) = x'_0$ for $x(t) \in \mathbf{R}^n, t \in [0, T]$
Explicit ODE Sensitivities:	Find $\frac{\partial x}{\partial p}(t)$ such that: $\dot{x} = f(x, p, t) = 0, t \in [0, T],$ $x(0) = x_0,$ for $x(t) \in \mathbf{R}^n, t \in [0, T]$
DAE/Implicit ODE Sensitivities:	Find $\frac{\partial x}{\partial p}(t)$ such that: $f(\dot{x}(t), x(t), p, t) = 0, t \in [0, T],$ $x(0) = x_0, \dot{x}(0) = x'_0,$ for $x(t) \in \mathbf{R}^n, t \in [0, T]$
Unconstrained Optimization:	Find $p \in \mathbf{R}^m$ that minimizes $g(p)$
Constrained Optimization:	Find $x \in \mathbf{R}^n$ and $p \in \mathbf{R}^m$ that: minimizes $g(x, p)$ such that $f(x, p) = 0$
ODE Constrained Optimization:	Find $x(t) \in \mathbf{R}^n$ in $t \in [0, T]$ and $p \in \mathbf{R}^m$ that: minimizes $\int_0^T g(x(t), p)$ such that $\dot{x} = f(x(t), p, t) = 0,$ on $t \in [0, T]$ where $x(0) = x_0$



## Model Evaluator : A More Advanced Optimization Example

---

### Equality and Inequality Constrained Optimization Under Uncertainty using Continuation:

Find  $x \in \mathbf{R}^n$  and  $p_1 \in \mathbf{R}^m$  that:

minimizes  $g_1(x, p_1)$

such that

$$f(x, p_1, p_2, p_3) = 0$$

$$g_2(x, p_1, p_2) = 0$$

$$g_3^L \leq g_3(x, p_1, p_2) \leq g_3^U$$

with uncertain variables  $p_2$   
and continuation parameters  $p_3$

- $N_p = 3$  parameter vectors:
  - design  $p_1$
  - uncertain  $p_2$
  - continuation  $p_3$
- $N_g = 3$  response functions:
  - objective  $g_1$
  - auxiliary equalities  $g_2$
  - auxiliary inequalities  $g_3$



## Model Evaluator Sensitivities

---

### First Derivatives

- State function state sensitivities:

$$W = \alpha \frac{\partial f}{\partial \dot{x}} + \beta \frac{\partial f}{\partial x} \quad [\text{LinearOpWithSolveBase}]$$

- State function parameter sensitivities:

$$\frac{\partial f}{\partial p_l}, \text{ for } l = 1 \dots N_p \quad [\text{LinearOpBase or MultiVectorBase}]$$

- Auxiliary function state sensitivities:

$$\frac{\partial g_j}{\partial x}, \text{ for } j = 1 \dots N_g \quad [\text{LinearOpBase or MultiVectorBase}]$$

- Auxiliary function parameter sensitivities:

$$\frac{\partial g_j}{\partial p_l}, \text{ for } j = 1 \dots N_g, l = 1 \dots N_p \quad [\text{LinearOpBase or MultiVectorBase}^2]$$

### Use Cases:

- Steady-state and transient sensitivity computations
- Optimization
- Multi-physics coupling
- ...



# Model Evaluator : “Composite” Coupled (Multi-Physics) Models

## Forward Coupled Model:

$$\tilde{f}^1(\tilde{x}^1, \tilde{p}_1^1) = 0$$

$$\tilde{p}_1^2 = \tilde{x}^1$$

$$\tilde{f}^2(\tilde{x}^2, \tilde{p}_1^2) = 0$$

## “Composite” Forward Coupled Model:

$$f(x, p_1) = 0$$

where

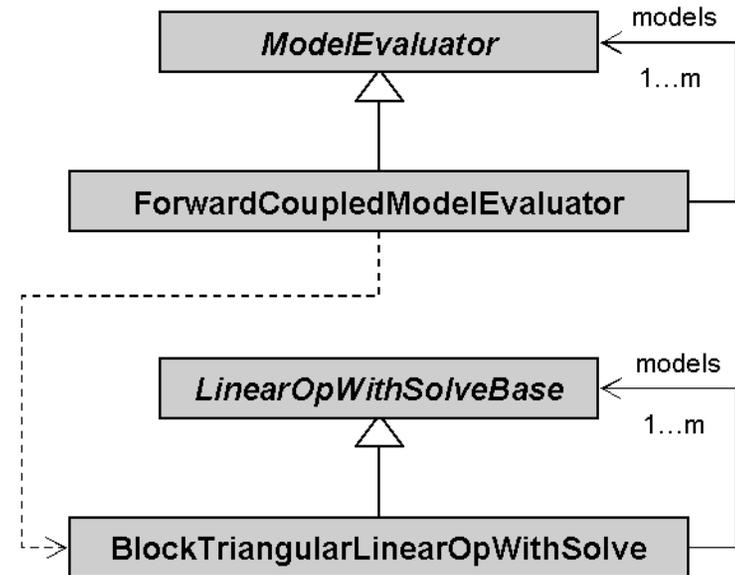
$$x = \begin{bmatrix} \tilde{x}^1 \\ \tilde{x}^2 \end{bmatrix}$$

$$p_1 = \tilde{p}_1^1$$

$$f(x, p_1) = \begin{bmatrix} \tilde{f}^1(\tilde{x}^1, \tilde{p}_1^1) \\ \tilde{f}^2(\tilde{x}^2, \tilde{x}^1) \end{bmatrix}$$

$$W = \beta \frac{\partial f}{\partial x} = \begin{bmatrix} \beta \frac{\partial \tilde{f}^1}{\partial \tilde{x}^1} & 0 \\ \beta \frac{\partial \tilde{f}^2}{\partial \tilde{x}^1} & \beta \frac{\partial \tilde{f}^2}{\partial \tilde{x}^2} \end{bmatrix}$$

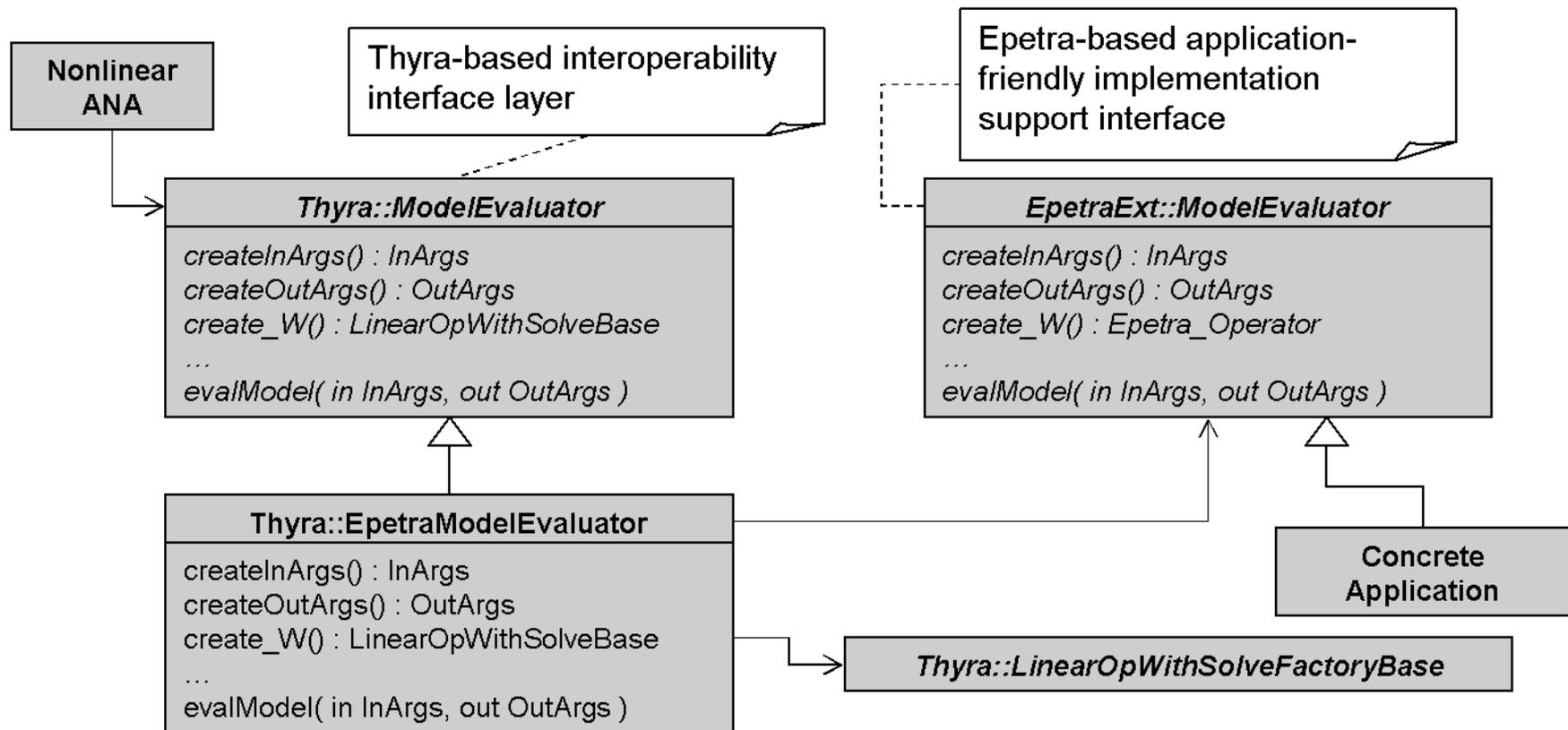
## “Composite” ANA Subclasses:



(Nonsingular)  
LinearOpWithSolveBase  
objects on the diagonal



## Model Evaluator : Thyra and EpetraExt Versions



- **Thyra::ModelEvaluator** and **EpetraExt::ModelEvaluator** are near mirror copies of each other.
- **Thyra::EpetraModelEvaluator** is fully general adapter class that can use any linear solver through a **Thyra::LinearOpWithSolveFactoryBase** object it is configured with
- Stateless model that allows for efficient multiple shared calculations (e.g. automatic differentiation)
- Adding input and output objects involves
  - Modifying only the classes **Thyra::ModelEvaluator**, **EpetraExt::ModelEvaluator**, and **Thyra::EpetraModelEvaluator**
  - Only recompilation of **Nonlinear ANA** and **Concrete Application** code



## Outline

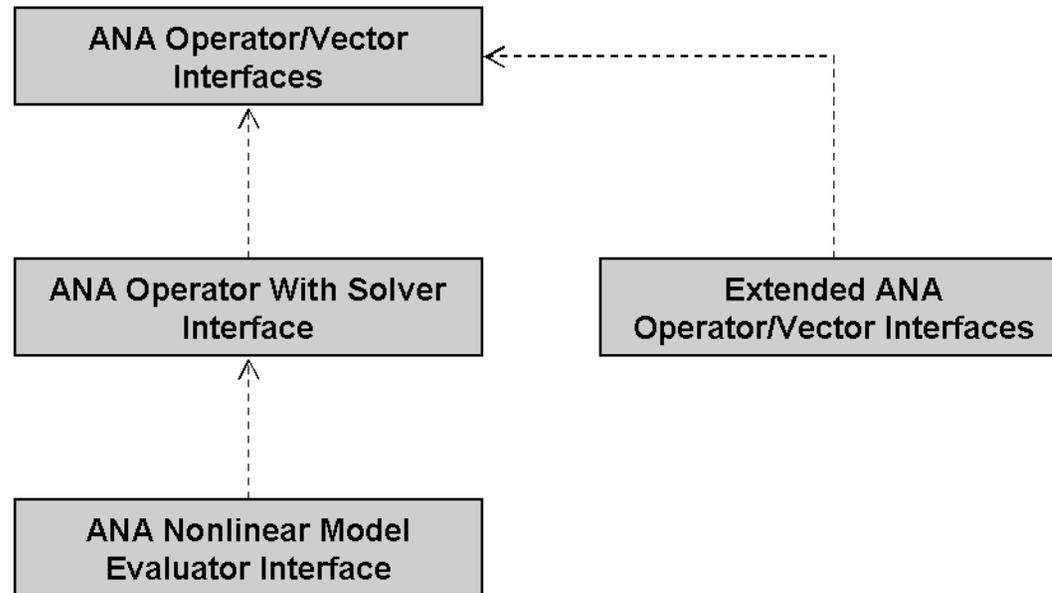
---

- Introduction of abstract numerical algorithms (ANAs) and Trilinos software and interfaces
- The need for package interoperability and layering
- Thyra requirements and the foundational ANA operator/vector Interfaces
- History behind of Thyra
- Current status of Thyra
- Prototype and future Thyra
- Wrapping it up



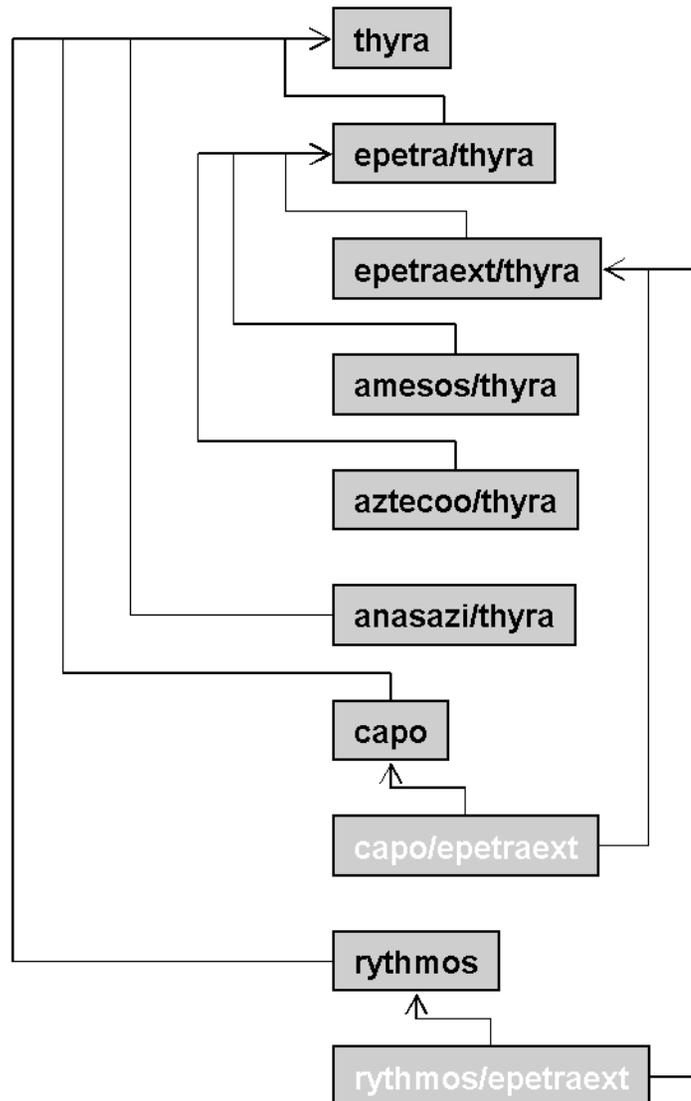
## Dependencies between Thyra Interfaces

---





## Trilinos Package Structure Related to Thyra



- The 'thyra' package appears early in the build process
- Thyra adapters are placed in the packages where the concrete implementations reside
  - Allows for scalable growth in the number of Thyra adapters
  - Encourages native package developers to take ownership of their Thyra adapters (Thanks Anasazi developers!)
- CAPO and Rythmos are written directly in terms of Thyra interfaces!
- **Guidance:** Ask me or another Thyra developer to review your Thyra adapters



# Thyra Doxygen Documentation

---

- Thyra Doxygen Documentation

- The main Thyra package documentation is broken up into different doxygen collections
  - Each doxygen collection should have an “Alphabetical List” page that fits in a single screen 1280 x 1064
- Documentation for Thyra adapters is find in the packages where they live (e.g. epetra/thyra adapers found in epetra) with links from main Thyra documentenation page.
- **Recommendation:** Build the doxygen documentation locally which enables source-code browsing

```
cd Trilinos/doc ; ./build_docs.pl
```

Open browser to `Trilinos/doc/index.html` [Link](#)

- **Recommendation:** Use the search features at <http://software.sandia.gov/trilinos> and at top of Doxygen pages
- **Caveat:** There are still typos, omissions, lies, etc ...



## Summary

---

- **Thyra** Interfaces provide minimal but efficient connectivity between ANAs and linear algebra implementations and applications
- **Thyra** is the **critical** standard for interoperability between ANAs in Trilinos
- **Thyra** can be used in Serial/SMP, SPMD, client/server and master/slave
- **Thyra** provides a growing set of **optional** utilities for ANA development and subclass implementation support
- **Thyra** support for nonlinear ANAs (i.e. model evaluator) is currently being developed as well as general support for linear solvers
- **Thyra** interfaces and adapters will be provided for preconditioner factories, implicit/explicit operator composition, and other features.
- **Thyra** adapters are available for Epetra, Amesos, AztecOO, Anasazi, and Rythmos with others on the way (e.g. Belos, NOX, MOOCHO ...)

### Trilinos website

<http://software.sandia.gov/trilinos>